User Guide
for
CSR1000/1 Development kits

Type: Draft

Version: 0.2

Release Date: Aug 17, 2011

Atlantik Elektronik GmbH
Address：Fraunhoferstr. 11a, 82152 Planegg, Germany
Phone：+49 (0) 89 89 505-0
Fax：+49 (0) 89 89 505-4134
http://www.atlantikelektronik.com

Release Record

Version Number	Release Date	Comments
0.1	Jul 18, 2011	First Draft Release
0.2	Aug 17, 2011	Second Draft Release - Added sections 3.3, 4.3.5

Table of contents

1. Introduction

This document is released for users and clients who are interested in CSR BTLE products and purchased CSR1000/1 Development Kits. It is intended to provide them with all necessary information needed for a faster development process, including descriptions on CSR low energy HW platforms, SW development environment and tools, and also a special part for SW developers which describes the basic CSR µEnergy code structure and programming tips with examples.

In order not to get confused when ordering a BTLE kit, the client should know that there are two variants that could be ordered, starting with DK and DB. DK stands for Development Kit, while DB stands for Development Board. A DK includes the programming board (CNS10020), mini-USB and mini-SMT cables, CSR1000/1 DB, CSR8510 USB-Dongle and the SW (µEnergy - SDK). So, the DK-CSR1000-10048 shall include the DB-CSR1000-10050-3a, while the DK-CSR1001-10049-2a shall include the DB-CSR1001-10051-2a. Note that both DKs could be used for programming both CSR1000 and CSR1001 chips, which means that if the user is still not sure which chip to use, he can purchase any DK and still be able to program both chips. To help the user decide which chip is more appropriate for him, it is worth mentioning that the main difference between CSR1000 and CSR1001 chips is the number of PIOs, which is 12 in the first case and 32 in the latter [1,2].

2. HW Platforms

2.1. CNS10020

Before describing the functionality of the CNS10020 board, it is worth mentioning that all CSR Bluetooth chips – BR/EDR and Low Energy – have two ports:

- UART: For communication with external MCUs and periphery
- SPI: For programming the Bluetooth chip itself (SPI_CLK, SPI_CSB, SPI_MISO, SPI_MOSI)

The CNS10020 board is used for programming the CSR1000 and CSR1001 chips, similar to the Casira-EP which is used for programming standard Bluetooth modules based on CSR chips. The CNS10020 board also works as a USB-SPI converter, which is connected to the user's computer USB port via a mini-USB cable. The board has also a micro-USB-like socket, called mini-SMT-connector that can be connected to either the DB-CSR1000-10050-3a or DB-CSR1001-10051-2a boards via a mini-SMT cable. The mini-SMT connector provides the DB boards with power (3V), UART pins and the four SPI pins, which are directly connected to the CSR1000 / CSR1001 chips on the DB boards. This is presented in figure 2.1. The CNS10020 board is illustrated in figure 2.2.

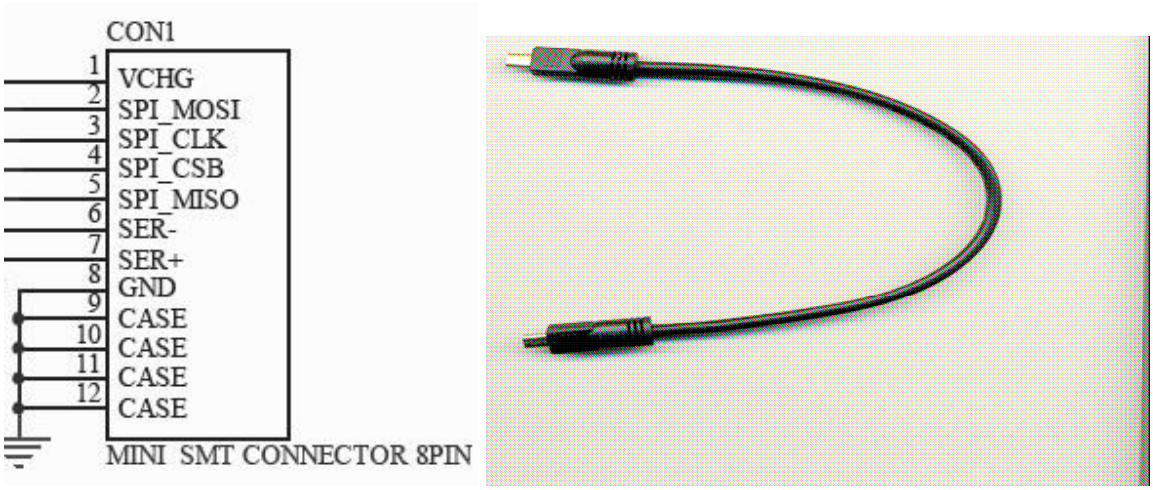

Figure 2.1: Mini-SMT Connector (left) and cable (right)

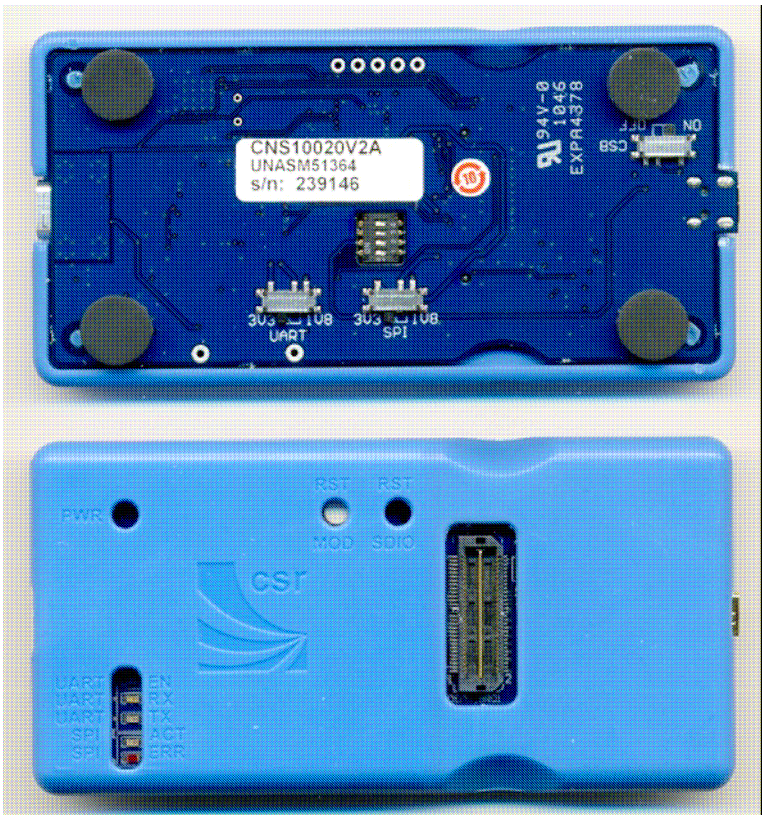

Figure 2.2: CNS10020 programming / converter board

In order to be able to use the CNS10020 board, a USB-SPI driver and FTDI virtual COM drivers need to be installed on the user's computer. The driver file is also shipped with the µEnergy SDK and can be found in the path "uEnergy_SDK-1.0\tools\bin\drivers" named "CDM20802_Setup.exe". The driver file is responsible for setting up the USB-SPI converter and also providing a virtual COM port for the USB-connected CNS10020 board. Note that the OS assigns a COM port number for this specific hardware, i.e. if the user changes the board, the new board shall be recognized as a new hardware by the system and shall be assigned a new port number. To check that the installation was successful, open the "device manager" and make sure to find the following lines:

- In "COM & LPT" -> "USB Serial Port (COMx)", where x is the COM number.
- In "USB Controller" -> "CSR USB<>SPI converter"

2.2. DB-CSR1000-10050-3a

The CSR1000 Tag board is ideal for simple sensor and proximity applications. It integrates the CSR1000 chip, external serial EEPROM, SMD antenna, buzzer, 2 LEDs, 1 push button, and access via connector to 3 AIOs and 1 PIO, in addition to the required external components like 16MHz and 32 kHz XTALs, as shown in figure 2.3.

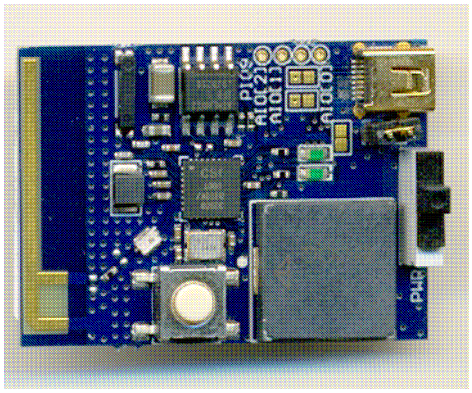

Figure 2.3: CSR1000 Tag board

2.3. DB-CSR1001-10051-2a

The CSR1001 development board is ideal for general purpose applications. It has no buzzer and no push button, but on the other hand it gives the user access to more interfaces compared to the CSR1000 Tag, like UART and more PIOs. It integrates the CSR1001 chip, external serial EEPROM, access via connector to 25 PIOs, 3 AIOs, and UART, in addition to the required external components like 16MHz and 32 kHz XTALs, external RF antenna socket, as shown in figure 2.4. The layout of the connector is illustrated in figure 2.5. Access to UART_Tx and UART_Rx pins can still be granted from the SER- and SER+ pins in the mini-SMT connector respectively, presented in figure 2.1.

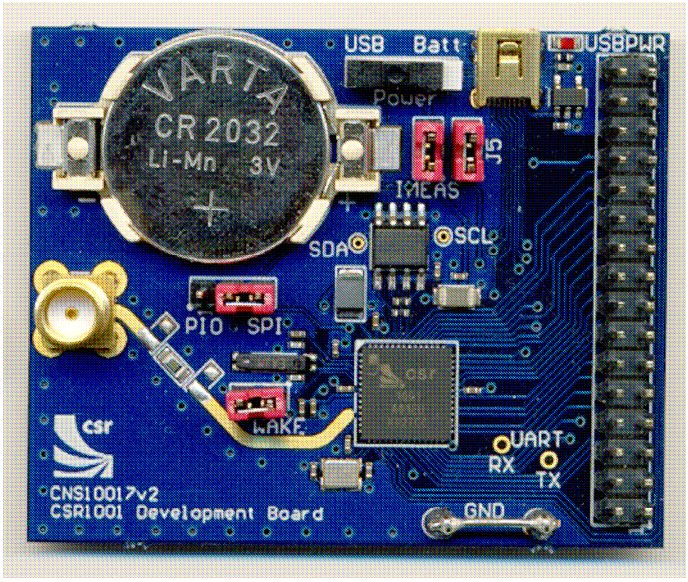

Figure 2.4: CSR1001 development board

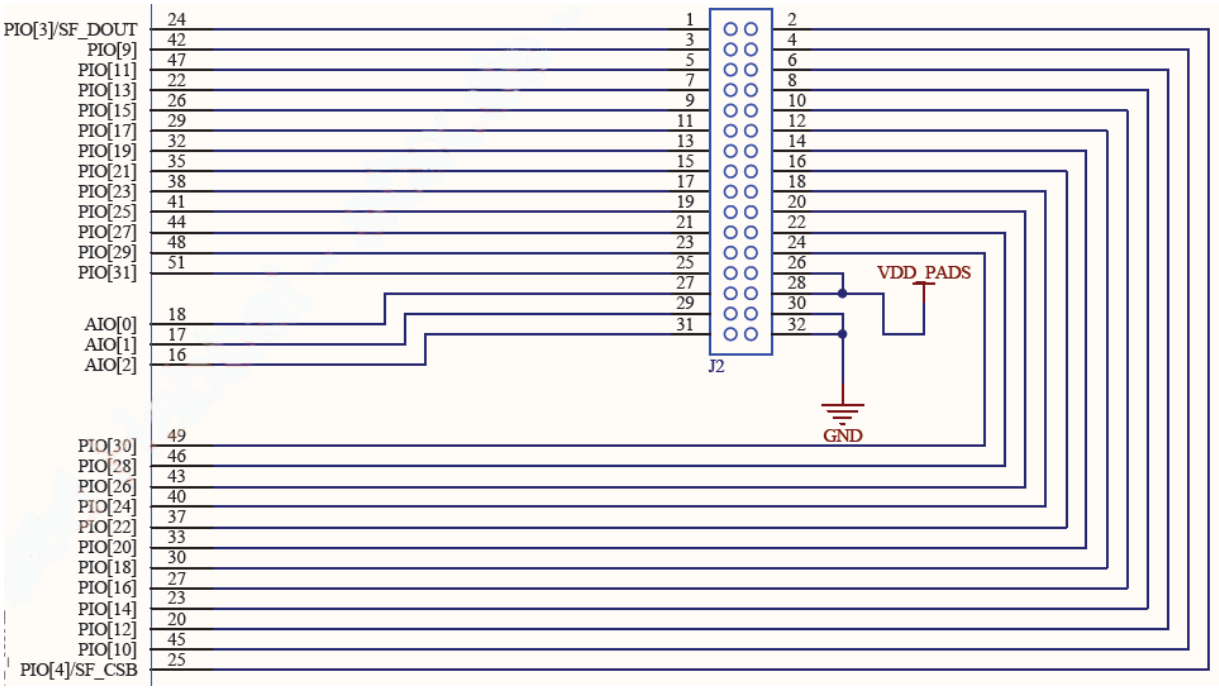

Figure 2.5: J2 connector pin layout for CSR1001 development board

3. CSR µEnergy SDK

This part is intended to provide developers with the information required to begin using xIDE to develop applications for CSR's Bluetooth single-mode low energy µEnergy ICs. Most of the information found in this part is referenced from the "CSR µEnergy xIDE User Guide". If the user has experience using BlueLab or BlueSuite SDKs, it would be very easy for him to use the µEnergy SDK and its tools.

3.1. CsConfig Tool

"CsConfig" is the main tool available in the SDK in addition to the xIDE programming environment. It is quite similar to "PSTool" in BlueLab or BlueSuite SDKs when dealing with traditional Bluetooth. The "CsConfig" tool provides the user access to different keys (parameters), like "Bluetooth Device Address", "Transmit Power Level", "UART Connection Speed", etc.
In addition to the traditional keys, CSR also provides a set of 8 unsigned integer "User Keys" (16-bit each), available for applications. A developer can set the keys to any value required, and read the values back via the CSReadUserKey() API [3].

When compiling a code on the xIDE and flashing the CSR1000/1 chips, the program resets all the keys to the default values, and the previous user settings are lost. In order to keep the desired keys, a "keyr" type file needs to be created and added to the project before compiling. To create such file after setting all the keys to the desired values, simply select "keys -> dump to file", and name the file after the projects name (i.e. enter <project name>.keyr), then save the file. More details can be found in section 3.1.2 in [3].
Note that the file name must not contain capital letters, or else it will not be recognized by the compiler and the merge process shall be ignored. To be sure the merge process was successfully done, make sure to find the following lines in the output window in the xIDE after flashing:

Download successful
C:/uEnergy_SDK-1.0/tools/bin/../bin/csconfigcmd.exe -trans 'SPITRANS=USB SPIPORT=0' merge <project_name>.keyr -reset
csconfigcmd, version 1.0
Copyright (C) 2010-2011, Cambridge Silicon Radio Ltd.
Success

3.2. µEnergy xIDE

"µEnergy xIDE" is quite similar to "BlueLab IDE" regarding the visual interface, main functions and short cuts like F7 for Build/compiling, F5 for Run/Debug, etc.
When creating a new project, two options are available: "blank project" and "minimal project". If the user chooses the "blank project", a directory shall be created with project name, containing 3 files: project file (.xip), workspace file (.xiw) and a makefile. The user then needs to add all necessary files by himself, like C-files, H-files and assembly files if needed. On the other hand, if the user chooses the "minimal project", the same process will occur, except that a 4th file will be created, named "main.c". This file is the main source file containing the minimal and basic "public functions" required for a project. More details to the basic source code structure can be found in section 4.2 in this document.

3.2.1. Creating a "Hello World" project

The best way to start using the µEnergy xIDE is by creating a "Hello World" project. As mentioned above, the simplest way to create such project is by choosing "Project -> New" then "minimal µEnergy project". The user does not need to integrate any code lines or libraries to see the "Hello World" output on the screen, it is all already integrated.
In the navigator window, under "C Files", the user can find the main.c source file. By double clicking, the code is opened in the main window. Instead of adding a library and function lines like the traditional "printf()", similar functions are implemented and used in the code, like:

```
DebugWriteString("Hello, world\r\n");
```

Which is found in the "debug.h" header file.

To see the output, press F7 to build the project then F5 to run and flash the board. On HyperTerminal (or other similar programs like HTerm, TeraTerm, etc) the output "Hello, world" can be seen, and confirms that the software has been installed and is working correctly. More details can be found in section 2.2.1.2 in [3]. Make sure to adjust the COM port number correctly to be identical to the COMx number shown in the Device manager as described in section 2.1.

What happens is that the CSR1000/1 chip sends the "Hello, world" string to the PC via UART (UART_TX pin). Both UART_TX and UART_RX pins are connected to the mini-SMT SER- and SER+ pins respectively (see figure 2.1). The CNS10020 board acts as a COM to USB converter and passes the data to the PC via the mini-USB cable. This means that the user can also see the same result if he uses an RS232 (serial COM) cable with an RS232 signal conditioner like MAX232 for example.

3.3. Compilation and flashing tips

Figure 3.1 illustrates the memory boot-up sequence for the CSR1000/1 chips. First, it checks for a valid ROM image to be copied to the RAM. If this is not found, it checks whether an EEPROM is connected to the I2C interface and whether it contains a valid image. If this is not found, it performs the last check on the SPI Master interface. More details about I2C and SPI Master interfaces can be found in sections 4.3.4 and 4.3.5.

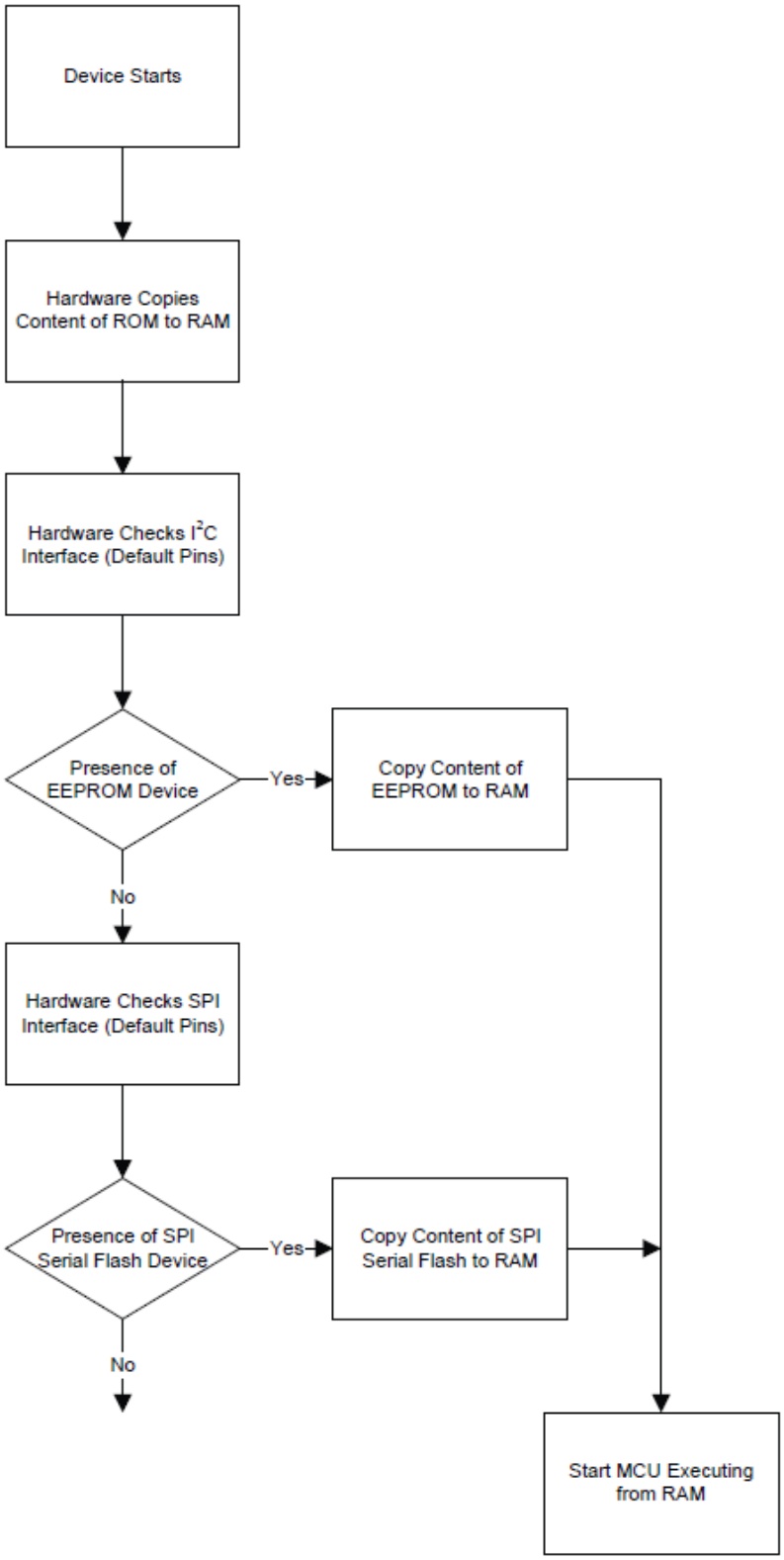

Figure 3.1: Memory Boot-up sequence [1]

Currently, CSR1000/1 chips are stand-alone chips and are shipped with a blank ROM. The BTLE mini-stack (L2CAP, GAP, ATT, GATT, SMP, etc) is shipped externally, presented in the “libgatt_lib.a” archive file. When compiling, the user application / project is linked with the archive files, and finally an image file (.img) is created to be flashed on the EEPROM. The whole build process is illustrated in figure 3.2.

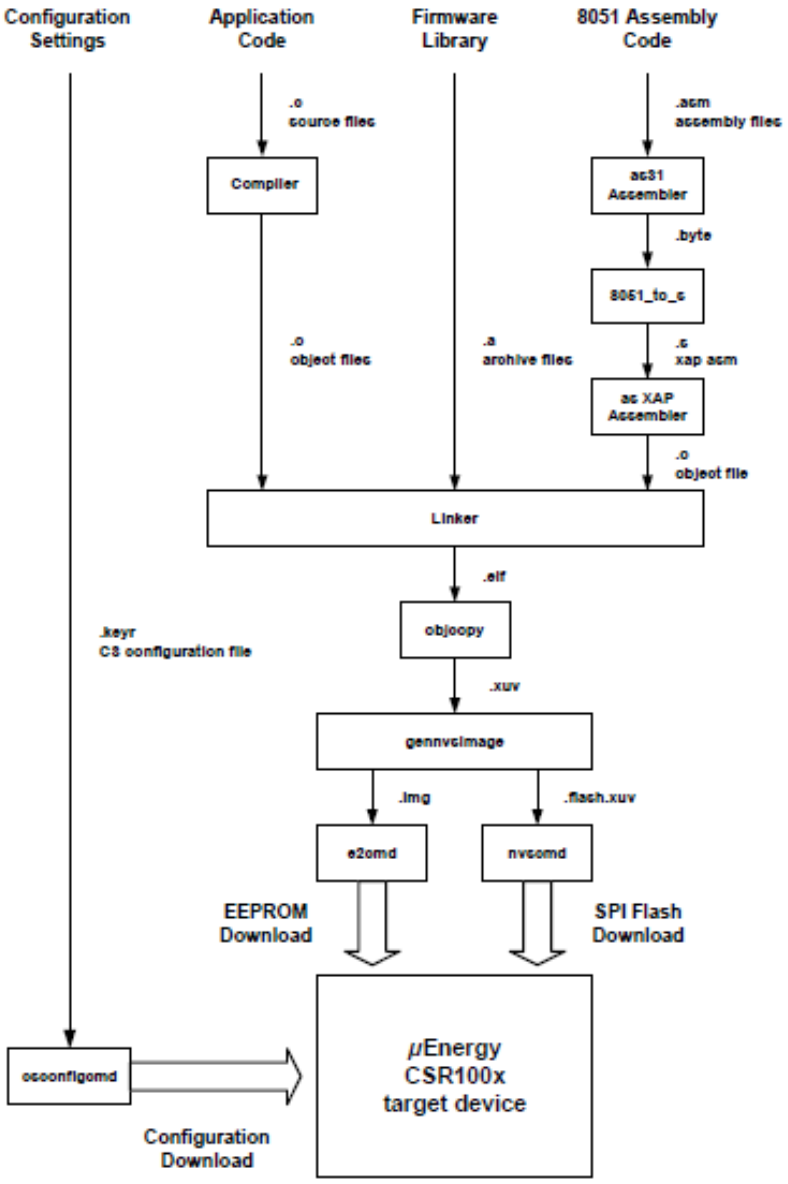

Figure 3.2: SDK build process [3]

4. SW Developer User Guide

This part is intended to provide SW developers with a better understanding of µEnergy source code basic structure and the main differences between traditional BT and BTLE Profiles. It also gives a basic description to CSR libraries and functions, in addition to CSR1000/1 chip related functions accessing chip peripherals and how to use them. At the end there is also a special part on how to implement a user specific Profile, but since this Profile will probably not be listed in the Bluetooth SIG Profiles list, it should be for internal use only and shall not be Bluetooth SIG compliant.

4.1. Differences between BT and BTLE Stacks

BT and BTLE stacks look quite similar regarding the basic structure and the usage of profiles, consisting of 3 main general layers; controller, host and application layers, as shown in figure 4.1. However, some new terms are defined for BTLE that do not exist for BT, while some got their names changed but still have the same functionality. Normally, BTLE modules on the market should support both the controller layers, which have the same names like in BT; PHY, LL and HCI, in addition to the host layers; L2CAP, ATT, GATT and SM. Unlike BT, security setting like authentications, pairing, encryption, etc. have been moved to a separate protocol in BTLE, which is the Security Management (SM). In traditional BT, all profiles supported on a BT device are listed in a special protocol called SDP (Service Discovery Protocol). Similarly, ATT/GATT protocols play the same role, except that each profile is sub-categorized into services, and each service has its characteristics. For example, the Proximity profile includes services like Device Information, Battery monitoring, Immediate alerting, Link Loss, Tx power, etc [5]. Some of these services are commonly used in several profiles. Each profile specifies which services are mandatory and which are optional.

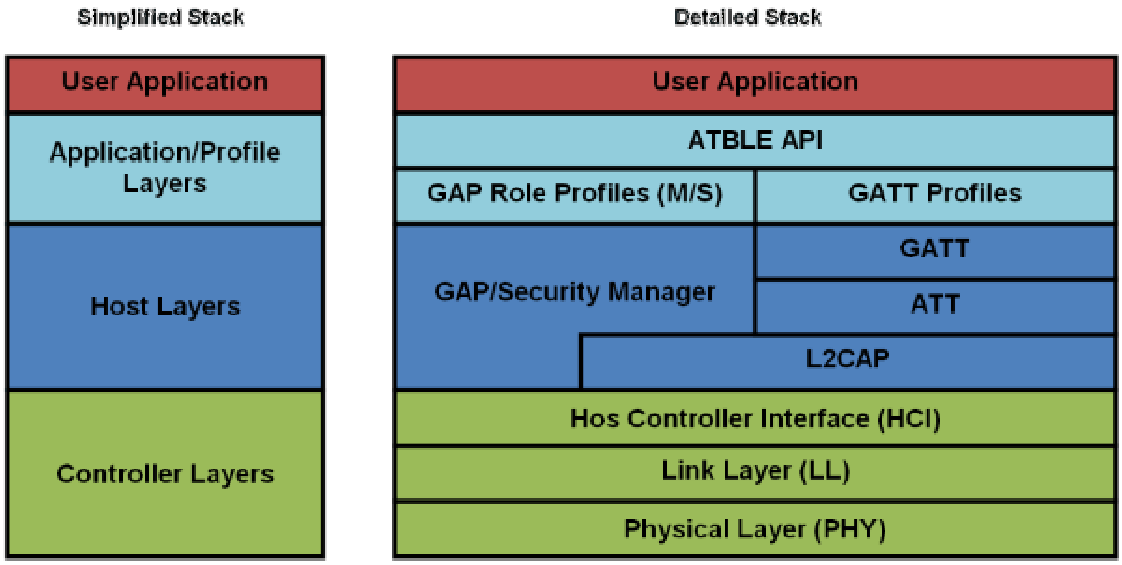

Figure 4.1: BTLE Stack

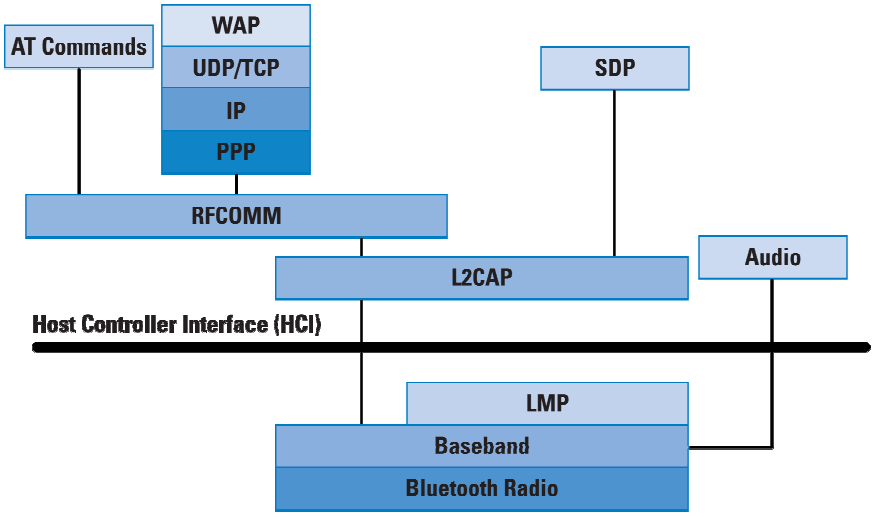

Figure 4.2: Sample of standard BT Stack

In BT, most profiles are implemented above intermediate layer protocols like RFCOMM, HID, AVDTP, BNEP, etc. Currently Bluetooth SIG finished specifying 4 profiles; PXP (ProXimity Profile), HTP (Health Thermometer Profile), HRP (Heart Rate Profile) and FMP (Find Me Profile). All these profiles lie above one protocol layer, which is the GATT. Some profiles are currently under development and shall be released Q3 or Q4 2011, like Simple Remote control Profile and HID Keyboard/Mouse Profile.
Access to the specifications of different profiles, services, protocols and core versions can be downloaded from the link in [6].

4.2. Basic source code structure

As mentioned in section 3.2, a BTLE project main source file should include 4 basic public functions:

4.2.1. *void AppPowerOnReset(void);*
This function is called just after a power-on reset. It is normally left blank.

4.2.2. *void AppInit(sleep_state LastSleepState);*
This function is called after the previous “AppPowerOnReset” function including system panic. Here initialization functions should be called. It is also called after wake up from deep sleep modes. CSR BTLE chips also provide the capability to protect some memory contents for some time (less than a minute) even after complete power cut off. This is very helpful in case battery is empty and the user needs to replace the battery. If the user managed to do that quickly, this function is called and reports the last sleep state of the chip.

4.2.3. *void AppProcessSystemEvent(sys_event_id id, void *data);*
This user application function is called whenever a system event, such as a battery low notification, is received by the system.

Four main system events can cause this function to be called, as described in the "sys_events.h" file:

sys_event_wakeup:
The system was woken by an edge on the WAKE pin.

sys_event_battery_low:
The system battery voltage is below the monitoring threshold.

sys_event_pio_changed:
One or more PIOs specified by PioSetEventMask() have changed input level.

sys_event_pio_ctrlr:
An event was received from the 8051 PIO Controller.

4.2.4. *void AppProcessLmEvent(lm_event_code event_code, LM_EVENT_T *event_data);*

This user application function is called whenever an LM-specific event is received by the system. This includes events generated from 4 main sources:

HCI Events
L2CAP Events
GATT Events
SM Events

All types of events generated and returned to this function are found in the "bt_event_types.h" file.

4.3. CSR chip related functions and interfaces

CSR1000/1 single mode chips support many interfaces like UART, I2C and SPI, besides a 16-bit ADC, timers, PWM etc. This section describes how to handle the chip related functions to control the on-chip HW and interfaces.

4.3.1. Timer

In any microcontroller, when a timer is started and the timer reaches the desired count, it enters the timer callback routine function to perform the required action. After that, the timers counter either resets and restarts or resets and stops, depending on the registers setup. For CSR BTLE chips, 2 functions are used to handle the timer; "*TimerDelete*" and "*TimerCreate*". In the timer callback function, if the user needs to stop the timer, the function "*TimerDelete*" should be called. If the user needs to restart the timer, again the "TimerDelete" function should be called, then the "TimerCreate" function hereafter. This is described in more details in the "timer.h" header file in the SDK.

4.3.2. UART

Several functions are implemented to support using the UART interface on the CSR BTLE chips, all could be found in the "debug.h" header file in the SDK, and are listed below:

```
void DebugInit(uint16 rx_threshold, uart_data_in_fn rx_event_handler,
        uart_data_out_fn tx_event_handler);
void DebugWriteChar( char const val );
void DebugWriteUint8( uint8 const val );
void DebugWriteUint16(uint16 const val);
void DebugWriteUint32(uint32 const val);
void DebugWriteTime48(uint16 const *val);
void DebugWriteString(const char *string);
```

The "*DebugInit*" function is used to initialize the UART interface and could be placed in the "*AppInit*" public function. If the user needs to send data values to an external device like MCU or sensor, then the function "*DebugWriteChar*" should be used to transmit the data in their original values without conversion. If the user needs to show the contents of a register for example on a HyperTerminal program, he could use the functions "*DebugWriteUint 8/16/32*" to convert the data into ASCII characters. The function "*DebugWriteString*" is used to transmit strings.
Note that the maximum baud rate while in Deep Sleep mode is 9600, based on the 32kHz crystal.

4.3.3. ADC / AIOs

CSR1000/1 has 3 AIOs, powered from VDD_AUX, which is an internal 1.35V supply voltage powered from the SMPS output. VDD_AUX is used as a reference voltage for all AIOs. Accordingly, the maximum analog input voltage level should be scaled down to a maximum of 1.35V.

4.3.4. I²C

The I²C interface communicates to EEPROM, external peripherals or sensors. An external EEPROM connection can hold the program code externally to the CSR1000 QFN. Figure 4.3 shows an example of an EEPROM connected to the I²C interface where I²C _SCL, I²C _SDA and PIO[2] are connected to the external EEPROM. The PIO[2] pin supplies the power to the EEPROM supply pin, e.g. VDD.
At boot-up, if there is no valid ROM image in the CSR1000 QFN ROM area the CSR1000 QFN tries to boot from the I²C interface. This involves reading the code from the external EEPROM and loading it into the internal CSR1000 QFN RAM.
Please note that the I²C interface can also be used to write user defined values during runtime on the EEPROM when connected. CSR is currently working on that and the feature shall be available in Q4 2011.

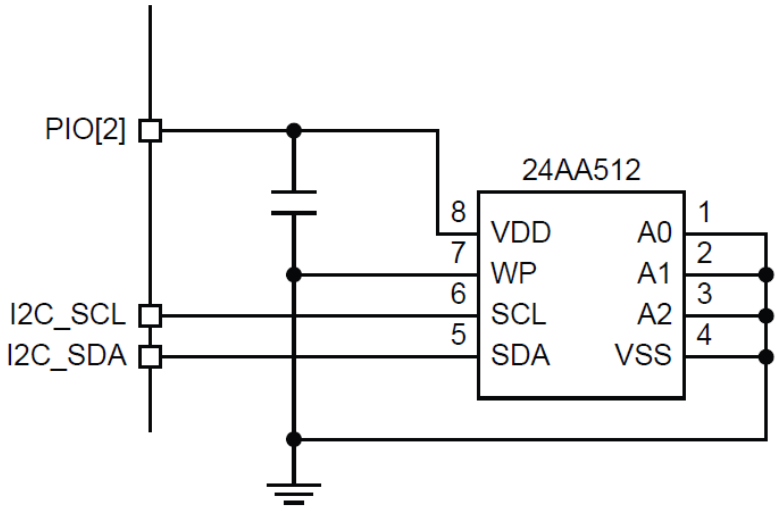

Figure 4.3: Example of I²C – EEPROM connection [1]

4.3.5. SPI Master

As mentioned in section 3.3, a serial Flash can be interfaced with CSR1000/1 chips via SPI master. The SPI master memory interface in the CSR1000 QFN is overlaid on the I²C interface and uses further 3 PIOs for the extra pins, see Table 4.1 below. If an application using CSR1000 QFN is designed to boot from SPI serial flash, it is possible for the firmware to map the I²C interface to alternative PIOs.

SPI Interface	Pin
Flash_VDD	PIO[2]
SF_DIN	PIO[3]
SF_CS#	PIO[4]
SF_CLK	I2C_SCL
SF_DOUT	I2C_SDA

Table 4.1: SPI Master Serial Flash Memory Interface [1]

4.3.6. PIOs

CSR1000/1 BTLE chips have the advantage that all PIOs could work as external interrupts. CSR1000 has 12 PIOs while CSR1001 has 32 PIOs. The PIOs could work in different modes as listed below, and all can be found in the “pio.h” header file in the SDK:

pio_mode_user (normal use as input or output pins)
pio_mode_edge_capture
pio_mode_pwm (set pin to PWM mode)
pio_mode_uart (Pins 0 & 1 are set to UART mode by default)

all other modes are found in the “pio.h” file. Other functions for setting pull-up / down, edge capture (rising, falling, both), set as input or output, etc. are found in the same header file.

Terms and Definitions

Terms	Definitions
EP	End Point
BR/EDR	Basic Rate/Enhanced Data Rate
BTLE	BlueTooth Low Energy
AIO	Analog Input Output
PIO	Programmable Input Output
SDK	Software Development Kit
RSSI	Received Signal Strength Indicator
SIG	Special Interest Group
SM	Security Management Protocol
ATT/GATT	Attribute protocol/ Generic Attribute protocol
GAP	General Access Protocol
SDP	Service Discovery Protocol
L2CAP	Logical Link Control and Adaptation Protocol
HCI	Host Controller Interface
PXP	ProXimity Profile
HTP	Health Thermometer Profile
FMP	Find Me Profile
HRP	Heart Rate Profile
ADC	Analog Digital Converter

List of Tables

List of Figures

References

1. CSR1000 QFN datasheet
2. CSR1001 QFN datasheet
3. CSR uEnergy xIDE User Guide
4. Bluetooth specification core version 4.0
5. Proximity Profile specification
6. https://www.bluetooth.org/Technical/Specifications/adopted.htm

Main Office:
Fraunhoferstraße 11a
82152 Planegg
Phone +49 89 89 505-0
Fax +49 89 89 505-100

Office Germany:
Deichstraße 17
20459 Hamburg
Phone +49 40 28 40 14-10
Fax +49 40 28 40 14-15

Office Benelux:
Industrieweg 3A
NL-4181 CA Waardenburg
Phone +31 418 57 23 80
Fax +31 418 65 33 22

Office Österreich:
Liebermannstraße A01 303-5
A-2345 Brunn am Gebirge
Phone +43 1 890 3703-10
Fax +43 1 890 3703-99

info@atlantikelektronik.com - www.atlantikelektronik.com